Flow to Interface the Camera Sensor

1) Interface the camera sensor correctly (see below) keeping the board off

To interface ArduCam

cs	c1	J6 pin 5
mosi	h1	J6 pin 4
miso	g1	J6 pin 1
sck	f1	J6 pin 3
gnd	Gnd	J7 gnd
vcc	3.3/5v	J7 vcc 3.3/5v
sda	sda	sda
scl	scl	scl

2) Turn on the board
3) Go to Vivado (/opt/Vivado/2023.2/bin/vivado) and flash with this bitstream : icarus v2.2.bit via hardware manager, ensure there is so negative slack
4) Go to /home/amrut/VEGA/vega-tools/utils/eth_transfer and do ./send.sh bbl.bin riscv.dtb (basically the buildroot)
5) Open fresh terminal and do minicom trisul32
6) Bootloader… do root x 2 and continue
7) Set up eth0 and enp1s0 everytime

On board :

```
ip link set eth0 up

 ip addr add 192.168.1.20/24 dev eth0
```

On Laptop :

```
sudo ip addr flush dev enp1s0

sudo ip addr add 192.168.1.10/24 dev enp1s0

sudo ip link set enp1s0 up
```

8) Write the required code in c, and to access the address, review the image in the next page…. (since camera is using i2c + spi, the image only describes flow for giving address for spi)
9) Scp from laptop the compiled capture binary (from /home/amrut/Downloads)

```
/home/amrut/VEGA/trisul32_buildroot/buildroot/output/host/bin/riscv32-linux-gcc
/home/amrut/Downloads/<whatever>.c -o <whatever> -static
```

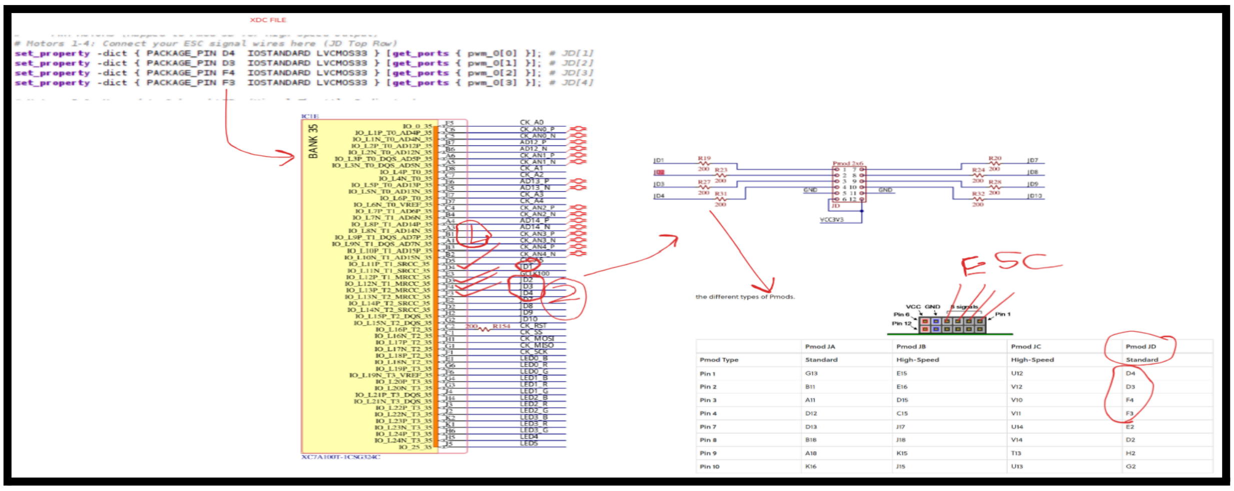
```
/home/amrut/VEGA/trisul32_buildroot/buildroot/output/host/bin/riscv32-linux-strip
<whatever>
```

```
ssh-keygen -f '/home/amrut/.ssh/known_hosts' -R '192.168.1.20'
```

```
scp -O <compiled_binary> root@192.168.1.20:/root/
```

10) Run the binary (ensure you strip the debug headers)
11) SCP the image.bin back
12) Run the python script as provided in /camera in git

To interface the camera sensor



```
# Motors 1-4: Connect your ESC signal wires here (JD Top Row)
set_property -dict { PACKAGE_PIN D4  IOSTANDARD LVCMOS33 } [get_ports { pwm_0[0] }]; # JD[1]
set_property -dict { PACKAGE_PIN D3  IOSTANDARD LVCMOS33 } [get_ports { pwm_0[1] }]; # JD[2]
set_property -dict { PACKAGE_PIN F4  IOSTANDARD LVCMOS33 } [get_ports { pwm_0[2] }]; # JD[3]
set_property -dict { PACKAGE_PIN F3  IOSTANDARD LVCMOS33 } [get_ports { pwm_0[3] }]; # JD[4]
```

	Pmod JA	Pmod JB	Pmod JC	Pmod JD
Pmod Type	Standard	High-Speed	High-Speed	Standard
Pin 1	G13	E15	U12	D4
Pin 2	B11	E16	V12	D3
Pin 3	A11	D15	V10	F4
Pin 4	D12	C15	V11	F3
Pin 7	D13	J17	U14	E2
Pin 8	B18	J18	V14	D2
Pin 9	A18	K15	T13	H2
Pin 10	K16	J15	U13	G2

To interface ArduCam

cs	c1	J6 pin 5
mosi	h1	J6 pin 4
miso	g1	J6 pin 1
sck	f1	J6 pin 3
gnd	Gnd	J7 gnd
vcc	3.3/5v	J7 vcc 3.3 -> 5v
sda	sda	sda
scl	scl	scl

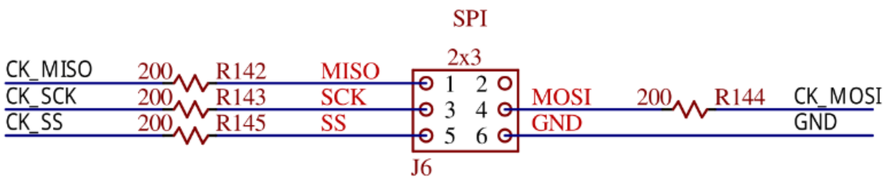
```
# --- SPI 1 (Camera/Aux) -> Mapped to Standard ChipKit SPI Header ---

set_property -dict { PACKAGE_PIN C1  IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_1_ss_io[0] }]; # CK_SS

set_property -dict { PACKAGE_PIN H1  IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_1_io0_io }];   # CK_MOSI

set_property -dict { PACKAGE_PIN G1  IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_1_io1_io }];   # CK_MISO

set_property -dict { PACKAGE_PIN F1  IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_1_sck_io }];   # CK_SCK
```



Why the board might not have internet………

Reasons:

1. FPGA Linux has **no Ethernet interface (`eth0`)**

2. No DHCP client

3. No default route

4. No DNS
5. Probably **no SSH server running**

Extra Information :) ---->

(In hardware and software, a magic number is a special, fixed value used for identification or control, often appearing as unexplained literals in code (e.g., 37px for positioning) or as file signatures (e.g., %PDF for PDFs) that uniquely identify file types or protocols, helping systems quickly recognize data formats. While useful for defining system parameters, they can make code harder to maintain, so they are often replaced by named constants for clarity, especially in hardware design and digital forensics

PNG File: Starts with 89 50 4E 47 (hex) which is ASCII for ".PNG".

JPEG File: Starts with FF D8 FF E0 (hex).

Why They Matter

Identification: Quickly identify file types or data formats without parsing the whole file.

Control: Dictate system behavior, like clock speeds in hardware designs.

Security: Vital in cybersecurity for malware detection and file validation.

Maintainability: Poorly documented magic numbers make code confusing; defining them as constants improves clarity)

Flow to Interface the mpu6050

- Interface the camera sensor correctly (see below) keeping the board off

To interface mpu6050

gnd	Gnd	J7 gnd
vcc	3.3/5v	J7 vcc 3.3v (5v)
sda	sda	sda
scl	scl	scl

- Turn on the board
- Go to Vivado (/opt/Vivado/2023.2/bin/vivado) and flash with this bitstream : icarus v2.2.bit via hardware manager, ensure there is so negative slack
- Go to /home/amrut/VEGA/vega-tools/utils/eth_transfer and do ./send.sh bbl.bin riscv.dtb (basically the buildroot)
- Open fresh terminal and do minicom trisul32
- Bootloader… do root x 2 and continue
- Set up eth0 and enp1s0 everytime
    - On board :
    - ip link set eth0 up
    - ip addr add 192.168.1.20/24 dev eth0
    - On Laptop :
        - sudo ip addr flush dev enp1s0
        - sudo ip addr add 192.168.1.10/24 dev enp1s0
        - sudo ip link set enp1s0 up
- Write the required code in c (we are only using the i2c), and to access the address, (in mpu6050-addrmapping) review the image in the next page…. (since mpu is using i2c)
- Scp from laptop the compiled capture binary (from /home/amrut/Downloads)
    - /home/amrut/VEGA/trisul32_buildroot/buildroot/output/host/bin/riscv32-linux-gcc /home/amrut/Downloads/<whatever>.c -o <whatever> -static
    - /home/amrut/VEGA/trisul32_buildroot/buildroot/output/host/bin/riscv32-linux-strip <whatever>
    - ssh-keygen -f '/home/amrut/.ssh/known_hosts' -R '192.168.1.20'
    - scp -O <compiled_binary> root@192.168.1.20:/root/

Our dts (device tree) file

```
/dts-v1/;
/ {
  #address-cells = <2>;
  #size-cells = <2>;
  compatible = "ucbbar,spike-bare-dev";
  model = "ucbbar,spike-bare";
  chosen {
    stdout-path = &SERIAL0;
    bootargs = "console=hvc0 earlycon";
  };
  cpus {
    #address-cells = <1>;
    #size-cells = <0>;
     timebase-frequency = <10000>;
    CPU0: cpu@0 {
      device_type = "cpu";
      reg = <0>;
      status = "okay";
      compatible = "riscv";
      riscv,isa = "rv32ima";
      mmu-type = "riscv,sv32";
      clock-frequency = <40000000>;



   CPU0_intc: interrupt-controller {
        #interrupt-cells = <1>;
        interrupt-controller;
        compatible = "riscv,cpu-intc";
      };
    };
  };
  memory@80000000 {
    device_type = "memory";
    reg = <0x0 0x80000000 0x0 0x10000000>;

  };
  soc {
    #address-cells = <2>;
    #size-cells = <2>;
    compatible = "ucbbar,spike-bare-soc", "simple-bus";
    ranges;
    clint@20010000 {
      compatible = "riscv,clint0";
      interrupts-extended = <&CPU0_intc 3 &CPU0_intc 7>;
  reg = <0x0 0x20010000 0x0 0xc0000>;

};

ethernet@20030000 {
            device_type = "network";
            compatible = "cdac,cdac_mac";
            interrupt-parent = <&plic0>;
            interrupts = <9>;
            reg = <0x0 0x20030000 0x0 0x100>;
     };

plic0: interrupt-controller@20010000 {
 #interrupt-cells = <1>;
 compatible = "cdac,plic-1.0.0";
 reg = <0x0 0x20010000 0x0 0x4000000>;
 riscv,ndev = <32>;
 interrupt-controller;
 interrupts-extended = <&CPU0_intc 9>;
 };

 SERIAL0: ns16550@10001000{
  compatible = "ns16550a";

  clock-frequency = <40000000>;

  reg = <0x0 0x10001000 0x0 0x100>;

  reg-io-width = <4>;

  reg-shift = <2>;

  current-speed = <115200>;


};

spi_clock:spi_clock{

        #clock-cells = <0>;

        compatible = "fixed-clock";

        clock-frequency = <40000000>;

    };

    spi0: spi@10000600 {

        compatible = "cdac,spi";

        clock-names = "spi_clock";

        clocks = <&spi_clock>;

        interrupt-parent = <&plic0>;

        interrupts = <6>;

        reg = <0x0 0x10000600 0x0 0x100 >;

        #address-cells = <1>;

        #size-cells = <0>;

        spidev@0 {
```

```
        compatible = "cdac,spidev";
        reg = <0>;
        spi-max-frequency = <25000000>;
    };
};
gpio0: gpio-controller@10080000 {
    compatible = "cdac,cdac-gpio";
    reg = <0x0 0x10080000 0x0 0x50000>;
    gpio-base = <0>;
    gpio-controller;
    #gpio-cells = <2>;
};
gpio1: gpio-controller@10180000 {
    compatible = "cdac,cdac-gpio";
    reg = <0x0 0x10180000 0x0 0x50000>;
    gpio-base = <16>;
    gpio-controller;
    #gpio-cells = <2>;
};
i2c0: i2c@10000800 {
            compatible = "cdac,mdp-i2c";
        clock-frequency = <100000>;
        system-frequency = <40000000>;
        #address-cells = <1>;

        #size-cells = <0>;

        reg = <0x0 0x10000800 0x0 0x100>;

        interrupt-parent = <&plic0>;

        interrupts = <8>;

        status = "okay";

};

    i2c1: i2c@10000900 {

        compatible = "cdac,mdp-i2c";

        clock-frequency = <100000>;

        system-frequency = <40000000>;

        #address-cells = <1>;

        #size-cells = <0>;

        reg = <0x0 0x10000900 0x0 0x100>;

        interrupt-parent = <&plic0>;

            interrupts = <4>;

        status = "okay";


};

pwm_clock:pwm_clock{

    #clock-cells = <0>;

    compatible = "fixed-clock";

    clock-frequency = <40000000>;
```

```
        };

        pwm0: pwm@10400000 {

            compatible = "cdac,cdac-pwm";

            clock-names = "pwm_clock";

                clocks = <&pwm_clock>;

            reg = <0x0 0x10400000 0x0 0x100>;

            #pwm-cells = <3>;


        };

    };

    };
```